Zielstellung

Dieser Schnelleinstieg demonstriert das Arbeiten mit dem Klassendiagramm in SiSy AVR, am Beispiel des myEthernet-Projektes.
Es werden alle Schritte gezeigt, die zur Erstellung eines Programms für einen Mikrocontroller notwendig sind. Zur Verwendung kommt die Projektvorlage myEthernet, in der bereits wichtige Klassen vorhanden sind.
Außerdem wird eine allgemeine Rahmenanwendung für Mikrocontroller vorgestellt.

Voraussetzungen

Für die Bearbeitung der Aufgaben benötigen Sie folgende Software und Hardware:

Software

- SiSy ab Version 2.18
- SiSy-Ausgabe AVR oder
  Developer, Professional bzw. BS
  mit dem integriertem ***Add-On AVR***
- Die myEthernet Projektvorlage

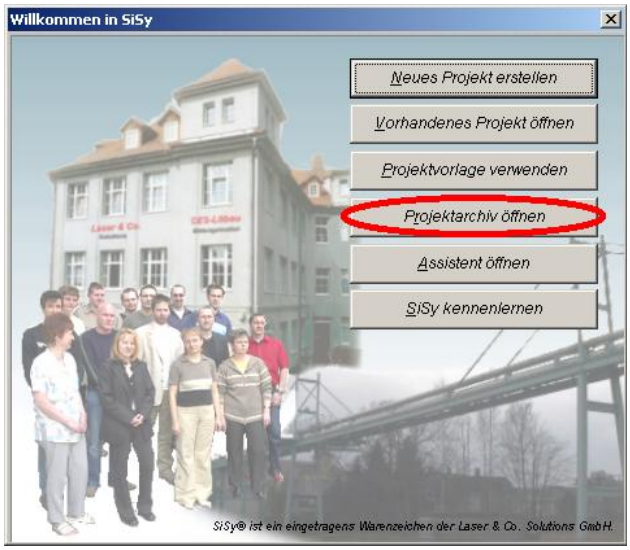
Hardware

- myEthernet Board
- ISP-Programmer für myEthernet

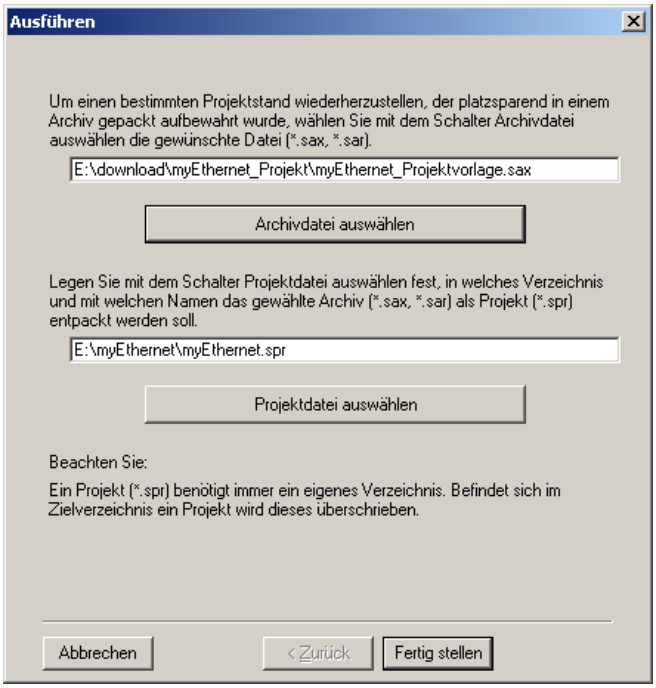
Als Programmer können Sie z. B. den mySmartUSB MK2 einsetzen

1. Ein neues Projekt anlegen

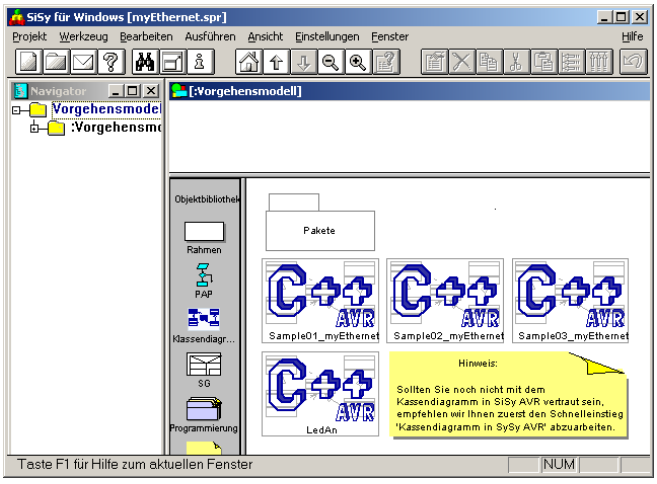
Starten Sie SiSy und wählen Sie im Startbildschirm *Projektarchiv öffnen*.



Im folgenden Dialogfenster laden Sie die Archivdatei *myEthernet_Projektvorlage.sax* und geben den Projektpfad an. Speichern Sie das Projekt auf jeden Fall in einem eigenen Verzeichnis.



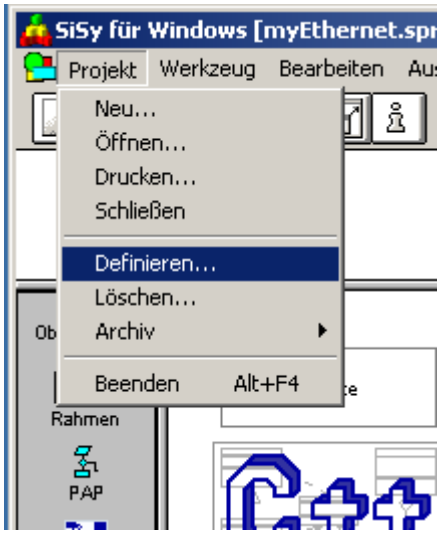
Das Projekt wird in der Ansicht *Vorgehensmodell* geöffnet. Dieses Diagramm bildet den Ausgangspunkt für die weitere Bearbeitung.



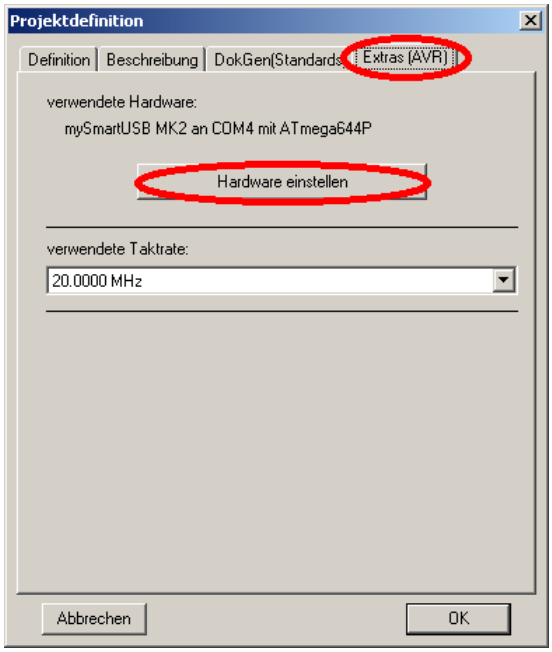
© Laser & Co. Solutions GmbH - 01/2012

2. Hardware einstellen

Als erstes sollte das soeben geladene Projekt an Ihre aktuelle Hardwarekonfiguration angepasst werden.
Öffnen Sie dazu den entsprechenden Dialog über die Menüpunkte *Projekt->Definieren...*



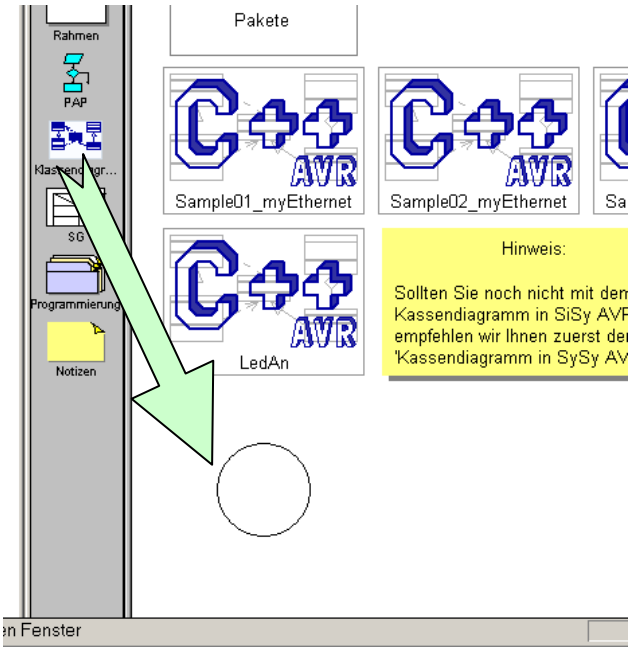
Wählen Sie die Registerkarte *Extras (AVR)* und dort die Schaltfläche *Hardware einstellen*.



Es öffnet sich das Programm zum Einstellen der Programmierverbindung. Stellen Sie Ihre vorhandene Programmierhardware und den Controller ein
(im Bild: **mySmartUSB MK2** und **ATmega644P**).

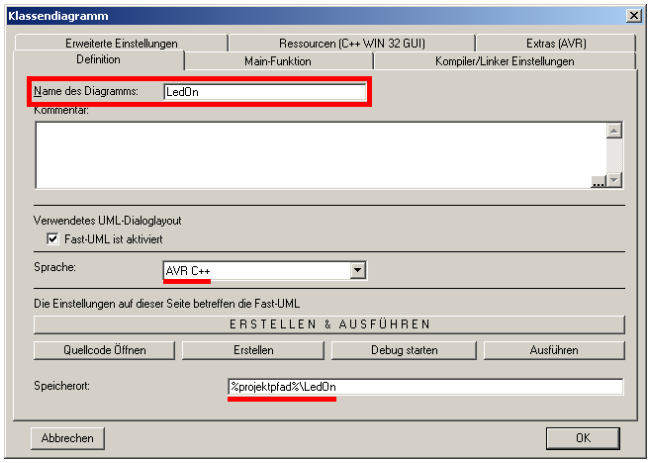
3. Neues Programm erstellen

Legen Sie ein neues Programm an. Ziehen Sie dazu ein Klassendiagramm aus der Objektbibliothek

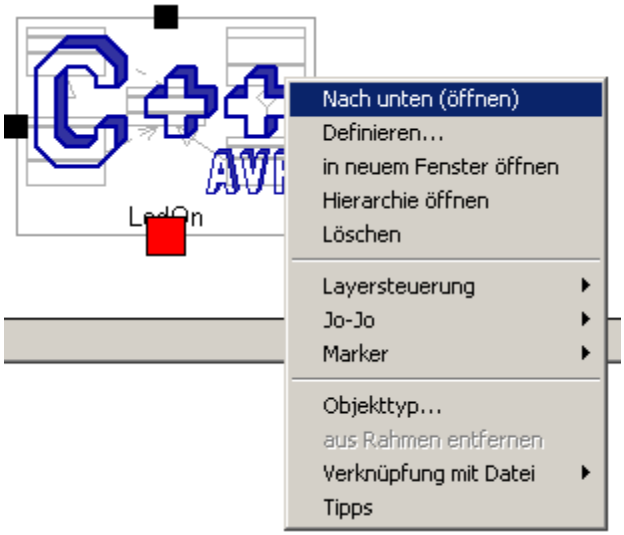


© Laser & Co. Solutions GmbH - 01/2012

Geben Sie dem Programm einen Namen. Achten Sie darauf, das als Sprache *AVR C++* eingestellt ist. Ein Unterordner für Ihr Programm wird automatisch angelegt.



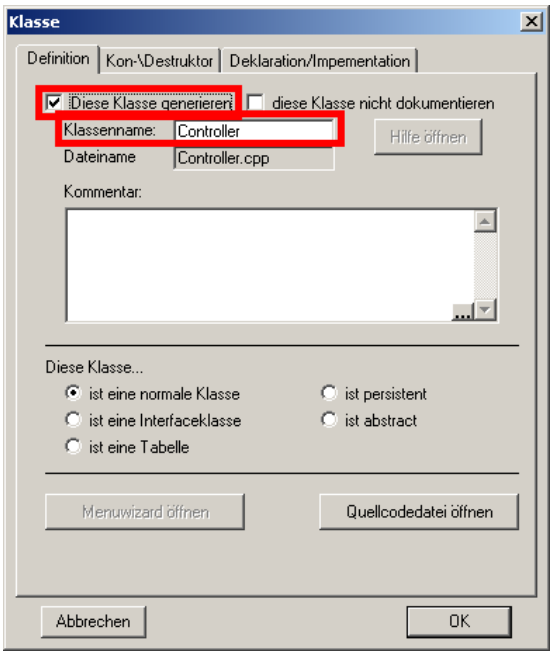
Wechseln Sie in Ihr neues Diagramm, indem Sie das Diagrammobjekt mit der rechten Maustaste anklicken und im erscheinenden Menü den Punkt *Nach unten (öffnen)* auswählen.



Im folgenden Dialog kann eine Programmvorlage gewählt werden. In diesem Beispiel sollen Sie jedoch alle Elemente selbst erstellen. Klicken Sie auf *Abbrechen*. Sie sollten jetzt das leere Klassendiagramm sehen.

4. Anwendung Grundgerüst

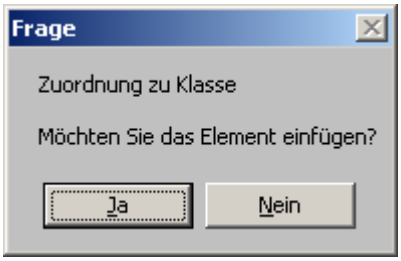
Als erstes wird ein Anwendungs-Grundgerüst erstellt. Dieses kann für jedes Mikrocontroller-Programm verwendet werden.
Ziehen Sie eine Klasse aus der Objektbibliothek in Ihr Diagramm. Diese Klasse repräsentiert das Programm auf dem Mikrocontroller.
Es öffnet sich ein Dialog, in dem Sie den Namen der Klasse festlegen. Setzen Sie außerdem den Haken in der Checkbox *Diese Klasse generieren*.



Der grundsätzliche Programmablauf besteht aus

1. Initialisierungen
2. Programmdurchführung (Main-Loop)

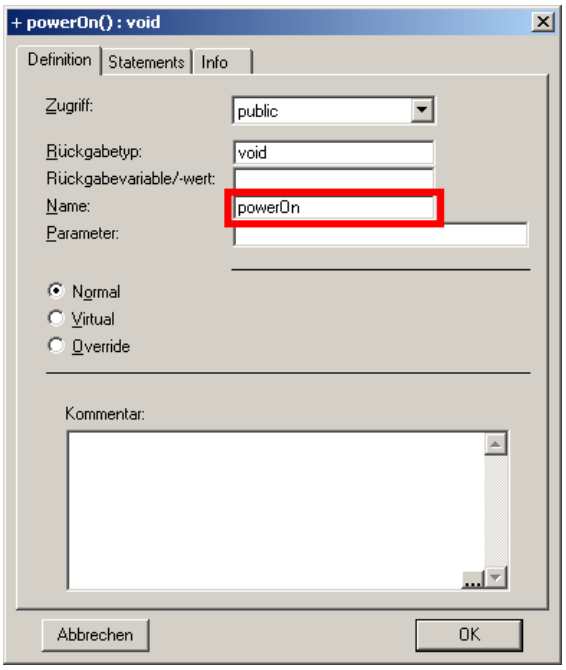
Ziehen Sie dazu nacheinander 2 Operationen aus der Objektbibliothek auf die Klasse *Controller*. Beantworten Sie die Fragestellung der MessageBox mit Ja.



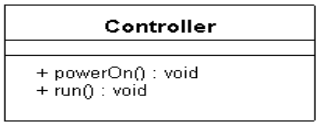
Sollte die MessageBox nicht erscheinen, haben Sie die Klasse nicht getroffen. Sie können die Operation später durch Verschieben auf die Klasse hinzufügen.

© Laser & Co. Solutions GmbH - 01/2012

Es öffnet sich der Dialog für die neue Operation. Legen Sie die Funktionsnamen *powerOn* und *run* fest.

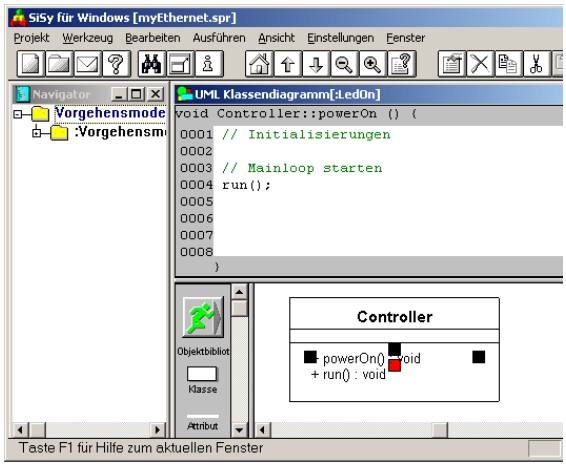


Die Klasse sollte jetzt folgendes Aussehen haben

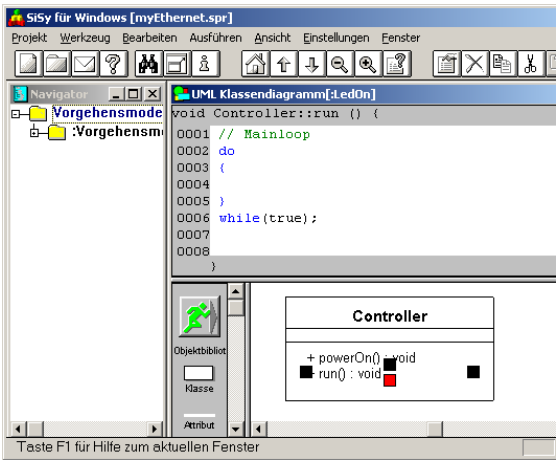


Wenn Sie eine der Operationen markieren, wird im oberen Teil des Diagramms der Quelltext dieser Funktion dargestellt. Mit der Taste F4 kann die Größe des Quelltextfensters umgeschaltet werden.
Der Programmablauf soll folgender sein.

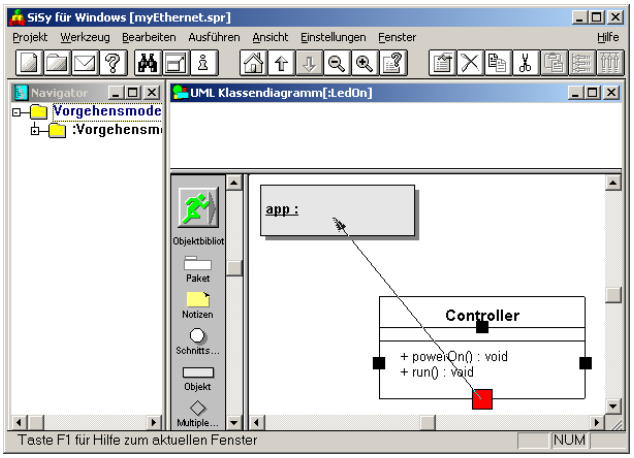
- Beim Programmstart (Reset oder Power On) wird die Operation *powerOn* ausgeführt. Hier werden alle Initialisierungen durchgeführt.



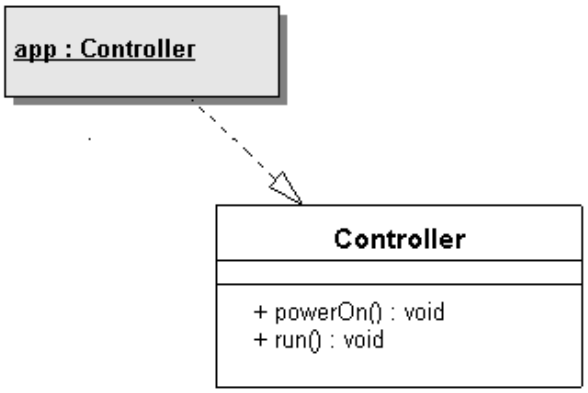
- Danach wird die Mainloop gestartet.



Um das Grundgerüst fertigzustellen, wird noch eine Instanz der Controller-Klasse benötigt. Ziehen Sie dazu ein Objekt aus der Objektbibliothek in Ihr Diagramm und geben Sie diesem Objekt den Namen *app*.
Markieren Sie die Klasse *Controller* (Achtung: nicht eine der Operationen sondern den Klassenrahmen markieren) und ziehen Sie eine Verbindung von dem unteren (roten) Marker der Klasse zum gerade angelegten Objekt *app*.



Bestätigen Sie den Dialog und der Klassenname sollte im Objekt eingetragen werden.
Das fertige Klassendiagramm des Grundgerüstes hat jetzt folgendes Aussehen:

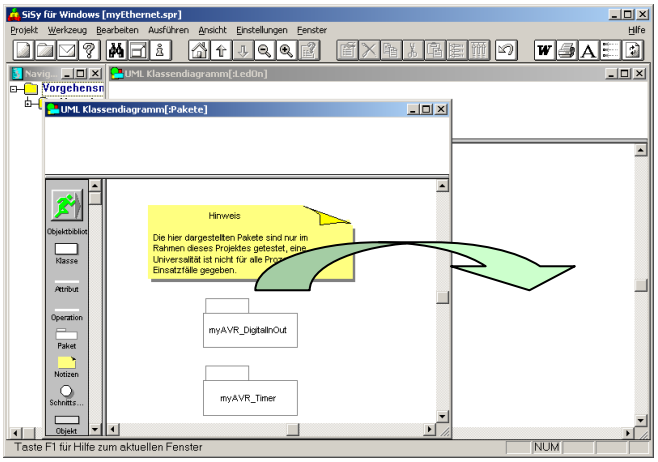


© Laser & Co. Solutions GmbH - 01/2012

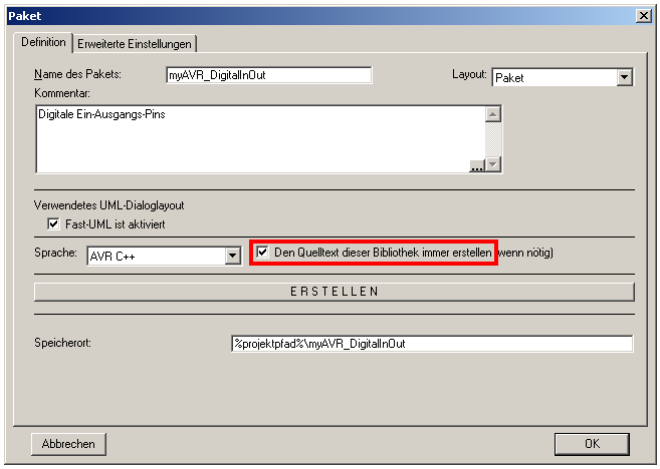
5. Bibliothek verwenden – LED einschalten

Als nächstes soll eine der Klassen aus den Bibliotheks-Paketen verwendet werden. Hier wird DigitalOut benutzt um die grüne LED zu schalten.
Öffnen Sie dazu ein neues Diagrammfenster.

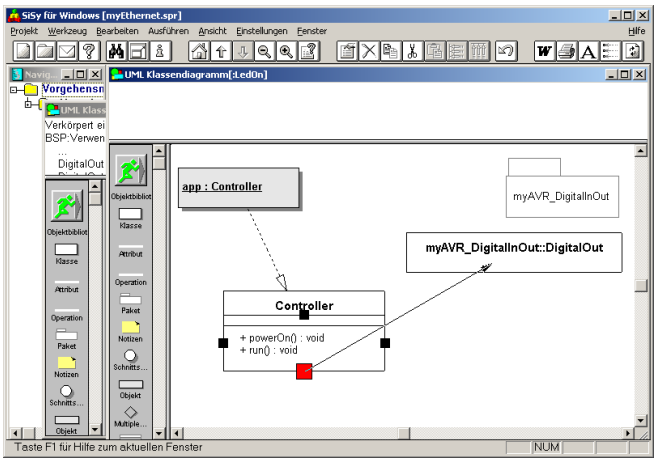
Wechseln Sie im neuen Fenster auf *Pakete* nach unten. Ziehen Sie das Paket *myAVR_DigitalInOut* in das Diagramm Ihres Programms



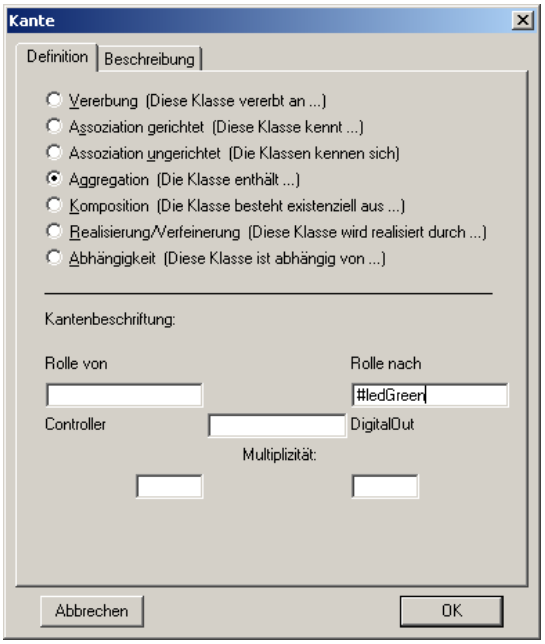
Wechseln Sie danach in das Paket (*Nach unten (öffnen)*) und ziehen Sie auch die Klasse *DigitalOut* in Ihr Programm.
Klicken Sie jetzt mit der rechten Maustaste auf das hereingezogene Paket *myAVR_DigitalinOut* und wählen Sie den Menüpunkt *Definieren ...* .
Es öffnet Sich ein Dialog, in dem Sie die Auswahlbox *Den Quelltext dieser Bibliothek immer erstellen* aktivieren müssen.



Erstellen Sie eine Verbindung von der Klasse *Controller* zu *DigitalOut.*



Wählen Sie den Verbindungstyp *Aggregation (Die Klasse enthält ...)* und tragen Sie im Feld *Rolle nach* ein: *#ledGreen* . Damit fügen Sie Ihrer Anwendung eine Instanz der Klasse *DigitalOut* mit diesem Variablennamen hinzu.



 © Laser & Co. Solutions GmbH - 01/2012

Die Klasseninstanz muss jetzt noch initialisiert werden. Dies erfolgt in der *powerOn*-Methode. Beim myEthernet ist die grüne LED mit Port C, Bit 4 verbunden.

UML Klassendiagramm[:LedOn]

```
void Controller::powerOn () {
0038 // Initialisierungen
0039 ledGreen.config(PORTC, BIT4);
0040 // Mainloop starten
0041 run();
0042
```

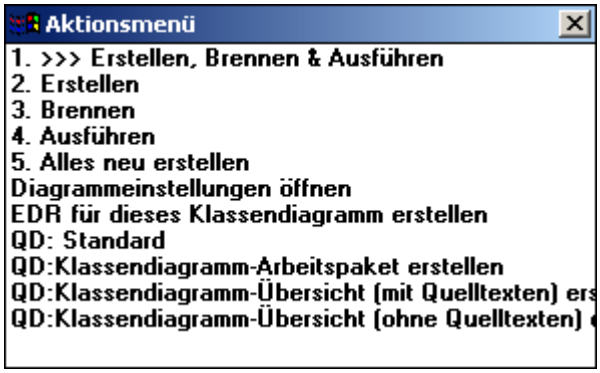
In der Mainloop, der *run*-Methode, kann die LED jetzt geschaltet werden.

UML Klassendiagramm[:LedOn]

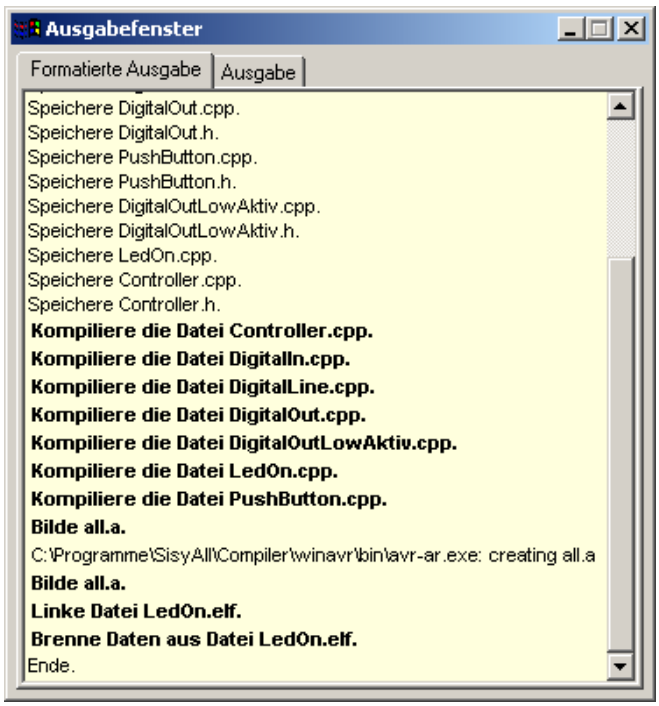
```
void Controller::run () {
0051 // Mainloop
0052 do
0053 {
0054     ledGreen.on();
0055 }
0056 while(true);
```

6. Erstellen und Brennen

Das Programm kann jetzt erstellt und gebrannt werden. Klicken Sie dazu auf die Schaltfläche für das Aktionsmenü in der Objektbibliothek.



Wählen Sie Punkt *1.*
>>> Erstellen, Brennen & Ausführen.
Beim ersten Erstellvorgang öffnet sich ein Dialog, in dem die Startfunktion festgelegt werden soll. Wählen Sie die Operation *powerOn*.



Beim Erstellen werden Dateien verwendet, die in diesem Diagramm nicht angelegt wurden (z.B. *PushButton.cpp*). Diese stammen aus dem Paket der Klassenbibliothek. Die Auswahl, welche Klassen vom Programm tatsächlich verwendet werden, erfolgt erst beim Linken. Das heißt, die zusätzlichen Klassen sind im fertigen Programm auf dem Mikrocontroller nicht enthalten.

Nach erfolgreichem Übersetzen und Brennen leuchtet jetzt die grüne LED Ihres myEthernets.

© Laser & Co. Solutions GmbH - 01/2012